Machine Learning for Physicists

Predicting Confirmed Cornona Cases Based on a Dataset of Population, Temperature and Air Pressure

Philipp Zolthoff
philipp.zolthoff@tu-dortmund.de

July 25, 2023

TU Dortmund – Fakultät Physik

Contents

1 Motivation

The last years were heavily defined by the devastating results of the COVID-19 pandemic [1], and we were greatly impacted by this virus. One of the most critical factors of any new disease is its growth, i.e., the confirmed cases in a given population. A common approach to analyzing this growth is to rely on exponential models for fitting the data. The main idea of this project is to develop an efficient way to predict the confirmed cases of COVID-19 patients using a selection of features that might correlate with the spread of the virus. As COVID-19 is primarily transmitted through everyday contact via aerosols [1], the selected features will focus on weather data, such as temperatures and air pressure. The goal is to investigate potential dependencies, such as how good weather might lead to an increase in outdoor activities, resulting in a higher spread of the virus. Conversely, bad weather might weaken the immune system, leading to increased testing and subsequent confirmation of cases. To isolate the problem from social status and other political factors, geological features will not be included in the analysis. To fully understand the impact of weather on COVID-19 cases, it is crucial to consider time dependencies. Thus, this project aims to train a recurrent neural network [4] to handle time-sequenced data and predict the confirmed corona cases of several cities. By incorporating temporal patterns, the model can better capture the dynamics of COVID-19 cases and improve prediction accuracy.

2 Used Datasets

The datasets used for this project include three main sets of data, providing information on a total of 8 features. Since the primary goal is to predict COVID-19 cases, the data must contain confirmed cases on the smallest possible geographical scale. Therefore, if only countries are compared, the data will be reduced to approximately 150 entries. For this project, the data will come from Johns Hopkins University, which provides a well-tracked ensemble of confirmed corona cases on a city scale.

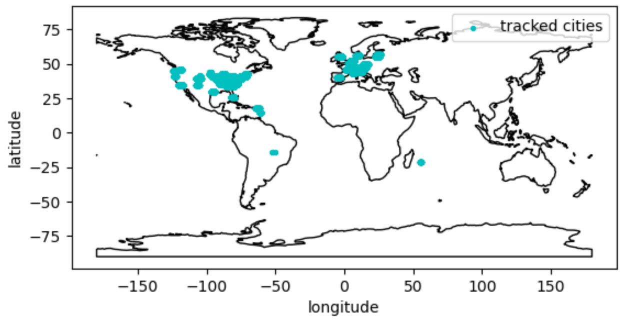


Figure 1: Shown is a schematic overview of the cities used to predict confirmed corona cases using GeoPandas [2].

Thus, an unprocessed amount of approximately 20,000 cities, coded through latitude and longitude, with the features of interest, is obtained. To collect worldwide weather data, again coded through latitude and longitude, the open-source platform “Metostat”

[3] will be used. They provide a vast amount of features, from which the project will use the air pressure, average, maximal, and minimal temperatures of each day. In a first step, both sets will be transformed into a “GeoPandasDataFrame”[2], which creates a new “Geometry” feature containing both longitude and latitude. With a chosen margin, in GeoPandas, this is called “buffer”, the entries will be joined with their representing counterparts that are close, i.e., within the margin of the first called buffer. For this, a buffer of 0.005 has been chosen to allow some deviation since the geological data points of the combined sets might not be taken at the exact same location. The final

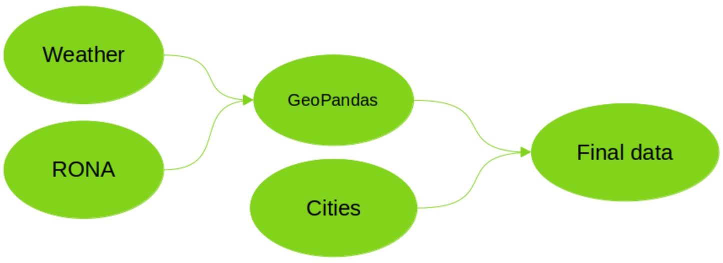


Figure 2: Shown is a schematic overview of how three different datasets are used to form one final set with the desired features.

DataFrame will include the features: confirmed cases, minimal temperature, maximal temperature, average temperature, air pressure, date, and population. Note that the date will be transformed into seconds, then divided by 10^{-8} rather than days, months, and years.

confirmed	tmin	tmax	tavg	pres	date	city	population
0	-5.5	5.0	-1.2	1030.9	1.579651	1294	10973.0

Figure 3: Exemplatory slice of the DataFrame used to predict confirmed COVID cases.

The scatterplot between the different features, including confirmed cases, can be seen in Figure 4. An obvious linear correlation between “T_min”, “T_max”, and “T_avg” can be observed. Furthermore, there is an almost exponential growth in the confirmed cases over time.

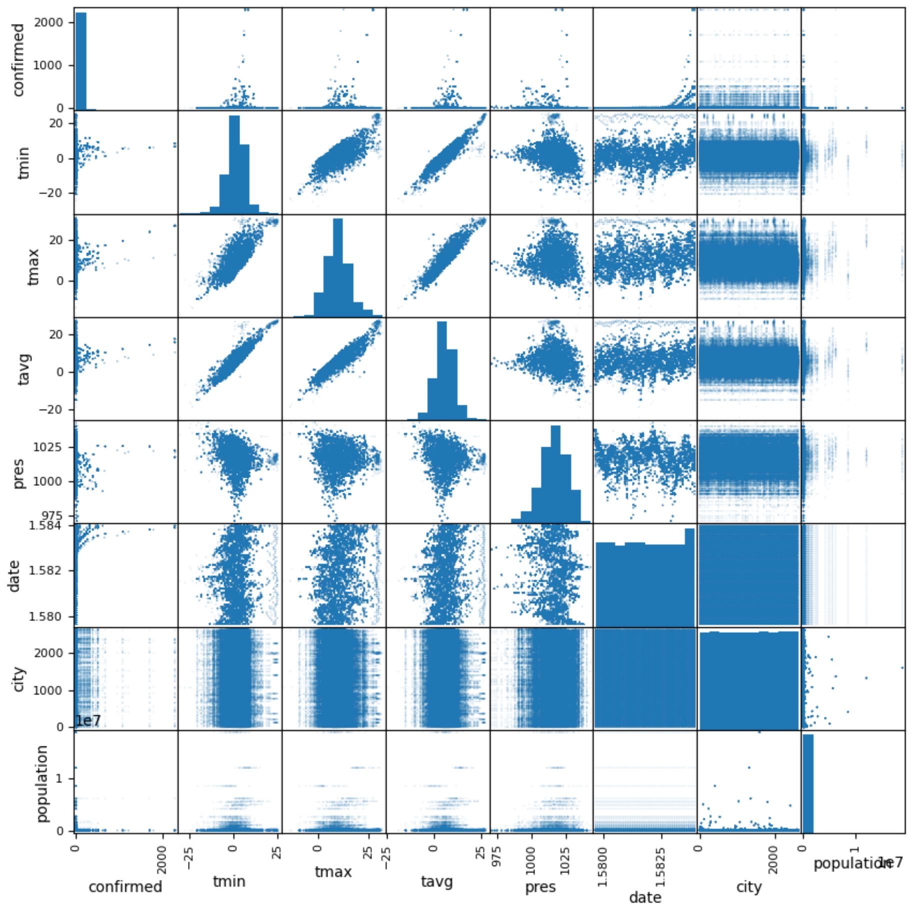


Figure 4: This graphic displays a scatter matrix of a dataset that contains several features related to both weather and confirmed COVID cases.

3 The Solution

3.1 Data Preperation

The problem that the neural network is trained for relies on time-dependent data, i.e., time-based weather data where the order of events has an impact on the outcome. To include that, recurrent neural networks, or RNNs for short, will be used as the foundation for a first approach. As an environment to build the model in, “Keras” has been chosen as it provides all the tools, layer structures, and evaluation methods to tackle the problem at hand [4]. The dataset needs to be further prepared and sorted so that the outcome shape will divide every city, with each city including 51 days of 8 features. This means that one entry contains 51 time-sequenced rows with information. A test-train split, provided by “Keras” [4], will randomly separate the data into different regimes: 20% for testing, 64% for training, and 16% for validation. Each regime will be split in two, where the first 40 entries, i.e. days, of each city will be the input of the model, and the last 11 entries of confirmed cases (shape (11,1)) will be the output, i.e., the target. To handle outliers and make the sets more manageable for the upcoming network, a “StandardScaler” [4] will be applied to each of them.

3.2 The Model

Since the already foreseen use of RNNs, three simple models have been constructed, each with one layer containing: “SimpleRNN”, “GRU”, and “LSTM”, to get a quick overview of how each RNN architecture performs on this problem.

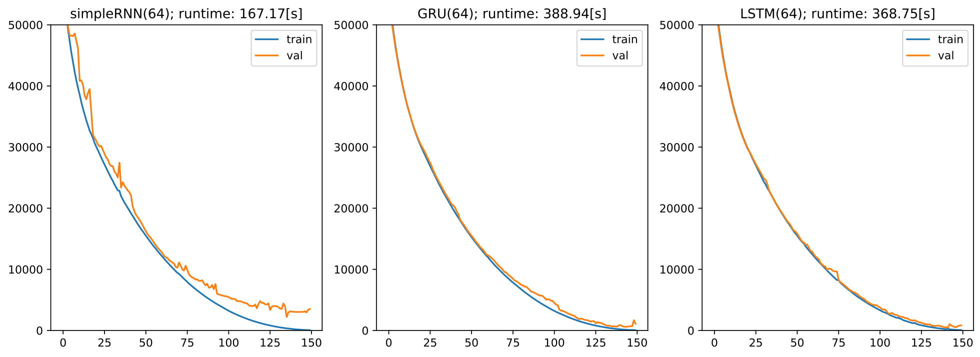


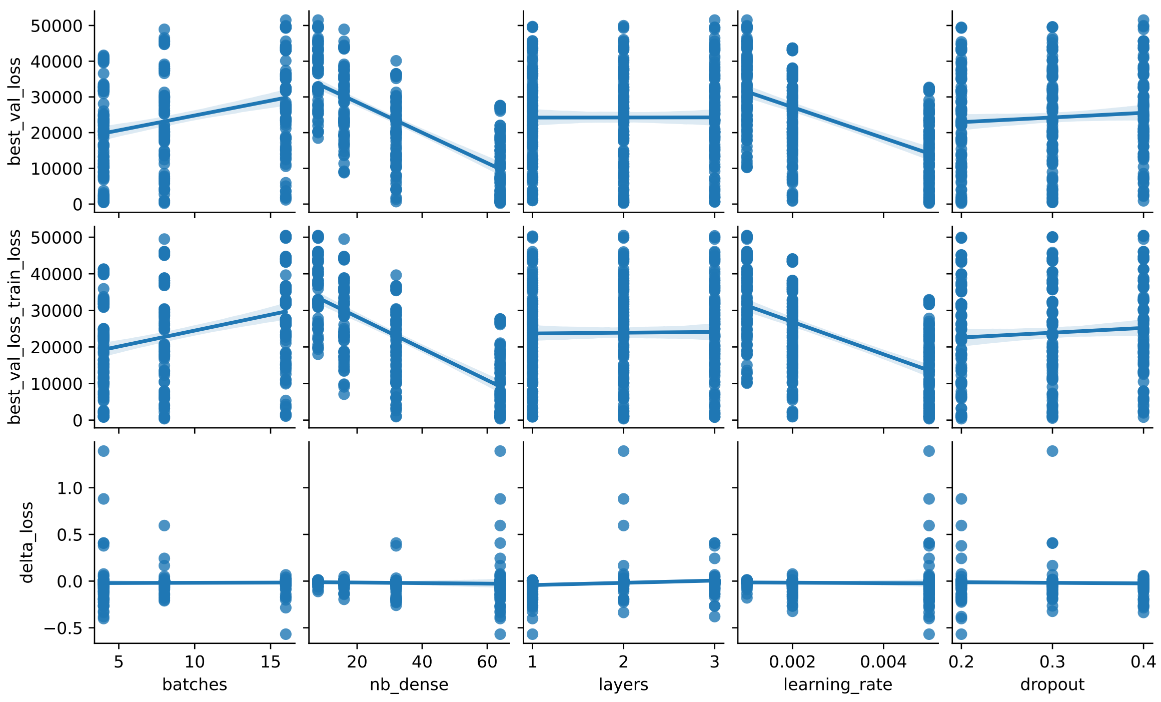
Figure 5: Displayed are three different RNN layers as loss functions, trained on the same dataset with the representing runtime in seconds.

The “SimpleRNN” loss function in 5 shows a dispersion between the train and validation data, indicating overtraining. However, it produces similar results to the more sophisticated layers. Both “GRU” and “LSTM” take approximately the same time for 150 epochs with similar results. In the following, “SimpleRNN” will be chosen as the main layer architecture for its short runtime, with “adam” as an optimizer [4]. This first selection has been done with only one layer as a simple representation.

3.3 Finetuning SimpleRNN

To apply a fundamental structure to the model, a grid search spanning over 324 combinations will look for the best outcome over an epoch count of 150, including an “early stop” mechanism with a patience of 10. The grid search will search for fitting candidates for: learning rate, units in each layer, layer count, dropout magnitude, and batch size. For this, each layer represents one layer of SimpleRNN architecture.

3.4 Gridsearch Results



	layers	nb_dense	dropout	learning_rate	batches	best_val_mae_train_mae	best_val_mae	best_val_loss_train_loss	best_val_loss	best_val_epoch
301	2	64	0.2	0.005	8	6.307665	4.993853	390.146729	244.596573	135
307	2	64	0.3	0.005	4	8.689884	6.171397	1119.690186	467.970978	92
310	2	64	0.3	0.005	8	6.820852	6.264912	556.710938	477.608124	126
298	2	64	0.2	0.005	4	10.660253	6.181417	975.801147	518.952271	97
308	3	64	0.3	0.005	4	8.314013	6.522310	838.901123	596.157043	102

Figure 6: Displayed is a hyperparameter optimization via gridsearch to find optimal parameters for: learning rate, dropout, numbers of layers and batch size. The top graphic displays a “pairplot” [5] inlcuding a linear fit to predict the models correlation towards a certain hyperparameter. The bottom grahpic shows the first five entries of the presented gridsearch, presenting the found parameters, sorted by “best_val_loss”.

The model picked for further finetuning is the third one at the bottom of 6, as it provides a good validation loss without a big discrepancy towards the training loss, indicating only low overtraining compared to different models found by the grid search. After the application of the grid search, some adjustments will be tried out manually, with a focus on the learning rate. For this purpose, a “learning rate scheduler” will be implemented as a further callback function that automatically activates itself after a given amount of epochs. The learning rate will then decrease in an exponential manner, and its argument can be further finetuned to achieve a smooth convergence towards the best possible loss rate. For the tuned model, an epoch of 100 will be chosen, with an argument of -0.05 in the exponential function, as from that point the validation loss struggles to converge. As the last layer is a dense model with 11 outputs (i.e., the 11 days the model has to predict), various scalers and output functions can be applied. Several scalers do not perform as well as the unscaled target in combination with a "softplus" (1) activation function, where

$$\text{softplus}(x) = log(exp(x) + 1). \quad (1)$$

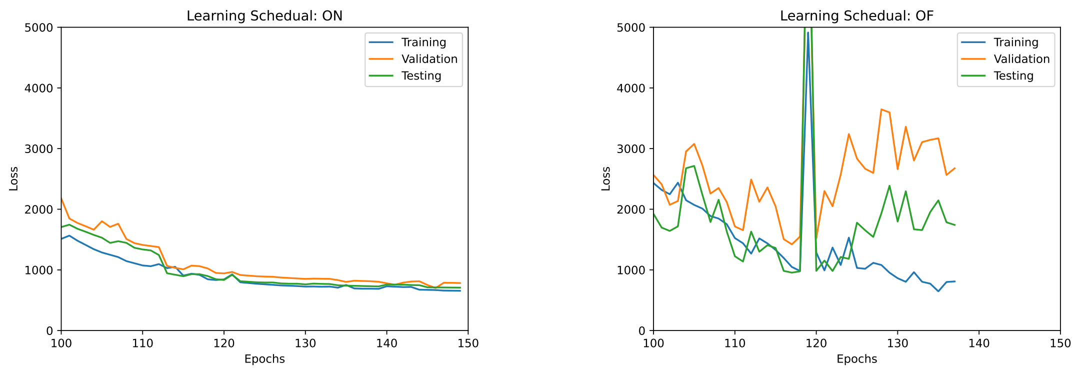


Figure 7: Displayed are two loss curves, both for the same dataset, trained on the same model, but with an activated learning scheduler on the left side. The right side displays the same model without the modified learning rate.

In 7, it can be seen how the loss function converges with a dynamic learning rate rather than a constant learning rate, where the supposedly local minimum is not found. The same results in the left graphic can also be achieved by lowering the constant learning rate to a certain amount, so that the minimum will be approached more carefully. However, this would require more epochs and thus more runtime to accomplish.

3.5 Keeping Overfitting in Check

To avoid overfitting, several mechanisms and imported tools will be applied to the model. An indicator of overfitting is when the validation loss converges, yet the training loss still decreases. To address this, the callback function "early stop" [4] will be applied with a patience of 10, which monitors the validation loss and stops training if the current validation loss is not undercut after 10 epochs. Furthermore, for each layer of SimpleRNNs, one layer of dropout will be added to force the units to be trained in a more generalized way, so that no single unit specializes too much for the training set.

4 Alternative Approach

As an alternative approach, the method of finding the least square for a given function will be chosen. With the approximately exponential behavior of most pandemics when it comes to confirmed cases, the fit will be applied to the form: $a \cdot \exp(b \cdot t) + c$, where t is the time in days, and a, b, and c are trainable parameters. Since the alternative method will not rely on a training dataset but on the 40 days ahead of the 11 that are about to be predicted, only the test data will be used here. For that, the first 40 days of every city will be fitted with an exponential function against the confirmed cases, followed by the calculation of the mean squared error of the then predicted following 11 days, to be later compared against the neural net. In this context, the alternative approach will rely on the size of the interval of the last days to be predicted. Increasing or decreasing the last 11 days as a target will thus make the resulting error smaller, not only because of fewer points to predict but also due to more data that trains the fit.

5 Results

The results contain both the model and a comparison of the model against the alternative approach. The goal of the model is to outperform the classic technique, not including neural networks.

5.1 Evaluation Of The Model

When the trained model is applied to the test data, it shows a final loss of 859.59 and a mean absolute error (MAE) of 8.43. Since the training process was monitored by the mean squared error (MSE) rather than the MAE, the MSE will be the leading factor in this evaluation. MSE has been chosen for its ability to include outliers more into its error in comparison with MAE. This characteristic, in this context, is more desired to suppress abnormalities, i.e. focus on the general increase of confirmed COVID cases.

5.2 The Model vs. The Alternative Approach

The model trained for this report and the alternative approach vary in both training concept and results. While the neural net profits from a test/train split to train on, the least-square-fit routine is directly applied to the first 40 days of the test dataset and thus does not experience the rest of the data. The MSE of the alternative approach reads 19944936.25. The figure in 5 shows the growing deviation of test data towards the end, i.e. the last days, where the exponential function strongly increases with every day. For the neural net already knows how the last 11 days can look like, because it was trained on that with the test and validation set, its error does not increase at a later period.

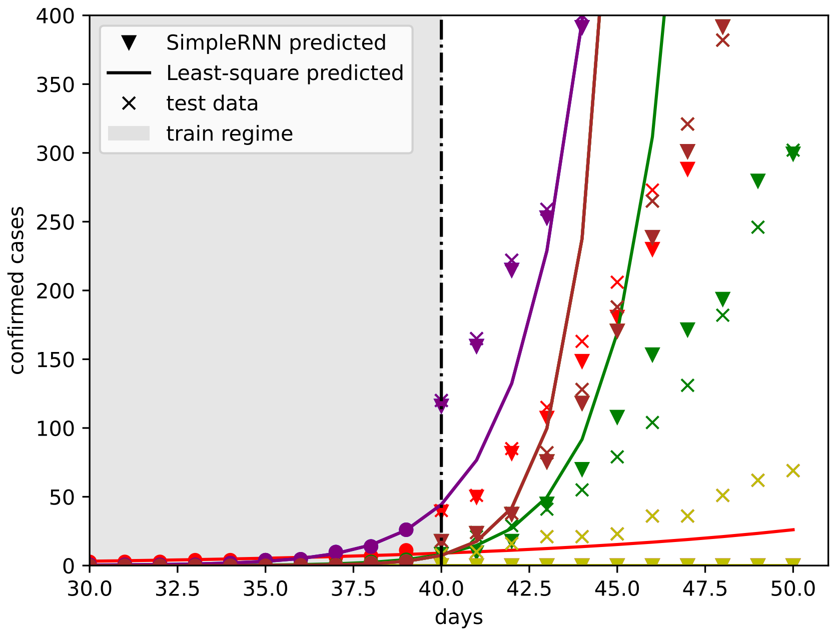


Figure 8: Displayed is a comparison of predictions on confirmed COVID cases between a least square fit routine and a trained neural net. The days are cut to the last 20 days, only including cities where confirmed cases have been found. Two different declarations of confirmed cases are given: SimpleRNN predictions and exponentialy fitted predictions, while the test data is marked with an x.

6 Conclusion

For the task of predicting confirmed COVID cases after a given 40 days of data, including weather information, the neural net outperforms the alternative approach by several magnitudes ($mse_{alternative} = 19944936.25 \quad \rightarrow \quad mse_{SimpleRNN} = 859.59$). The alternative method heavily relies on the amount of days it is fitted with, while the SimpleRNN seemingly knows how to predict a certain course based on the weather and confirmed cases before the prediction. Even though the neural net performs better, it still gets some predictions wrong, as can be seen for the yellow case in 8. Another approach to the model would be to train the data on a more sophisticated RNN architecture, for example, “LSTM”, which would take more time and resources but might perform better on the problem at hand.

References

[1] Ensheng Dong, Hongru Du, and Lauren Gardner. “An interactive web-based dashboard to track COVID-19 in real time.” In: *The Lancet Infectious Diseases* 20.5 (2020), pp. 533–534. ISSN: 1473-3099. DOI: `https://doi.org/10.1016/S1473-3099(20)30120-1`. URL: `https://www.sciencedirect.com/science/article/pii/S1473309920301201`.

[2] Kelsey Jordahl et al. *geopandas/geopandas: v0.8.1*. Version v0.8.1. July 2020. DOI: `10.5281/zenodo.3946761`. URL: `https://doi.org/10.5281/zenodo.3946761`.

[3] Christian Sebastian Lamprecht. *Meteostat Python*. URL: `'https://orcid.org/0000-0003-3301-2852'`.

[4] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[5] Michael L. Waskom. “seaborn: statistical data visualization.” In: *Journal of Open Source Software* 6.60 (2021), p. 3021. DOI: `10.21105/joss.03021`. URL: `https://doi.org/10.21105/joss.03021`.

Appendices

Code With Attached Link To Google Colab

building and training the model:
`https://drive.google.com/file/d/1g98-K3Ew38aHmZI_UjKjSJ5oHohP6wZn/view?usp=sharing`

building the alternative approach and comparing it to the model:
`https://drive.google.com/file/d/19wJ7KO5qzxced04jpotOsPgFXzI9ch9u/view?usp=sharing`

building the dataset:
`https://drive.google.com/file/d/1nuwFR147KMsDrmzwjBTYvJ3D25ndMkZ-/view?usp=sharing`